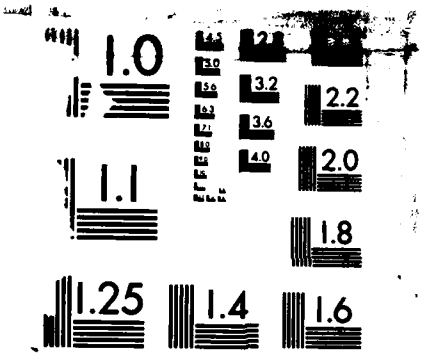
AD-A174 600 THE DESIGN AND IMPLEMENTATION OF A NETWORK COMPUTER(U) 1/1
STATE UNIV OF NEW YORK AT STONY BROOK A J BERNSTEIN
29 MAY 86 AFOSR-TR-86-0783 AFOSR-81-0197

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART

STANDARDS 1963-A

Final Research Summary
AFOSR81 0197
Professor Arthur J. Bernstein

The research performed under this grant is concerned with distributed languages and algorithms. The results obtained can be divided into 5 major areas:

A. Distributed Languages - High Level Considerations

The concern here was to design a distributed language which has the property that the modules of a program written in it can be distributed over the nodes of a network computer. The correctness of the program must not depend on how the distribution is performed, although clearly its run time efficiency will be effected. Such distributed languages have been investigated by a number of researchers but our view of the problem is somewhat different. We are interested in languages that support low level, systems applications which rely on flexible features that can be efficiently implemented and that do not constrain the programmer in the way an algorithm is formulated. Such algorithms are of interest in operating system design and are the subject of a growing body of research literature. Examples include election algorithms, distributed deadlock detection algorithms, algorithms for achieving distributed synchronization and load balancing and algorithms for achieving reliable operation in an unreliable environment.

Algorithms of this type may involve one or more of the following features: replication of information, redundant computation, resiliency in the face of inconsistent information, communication failures or node failures. These algorithms are generally characterized by a rather high level of message type communication between distributed processes. Processes generally do not wait for a response immediately after sending a message and in many cases there is no response at all. Furthermore, communication is frequently of a multicast or broadcast nature. Multicast is particularly significant in applications concerned with distributed synchronization and reliability and is naturally supported by current communication technology (e.g., Ethernet). Therefore, multicast features in a language can be efficiently implemented. Another aspect of communication in such algorithms is that a message need not always be addressed to a unique process; any one of a set of processes may be eligible to receive it. For example, for reasons of reliability, load sharing or to reduce the lengths of communication paths, duplicate servers may be distributed throughout a network. In such situations the destination of a message cannot be determined at compile time since there may be several processes which are equally suitable as targets. The decision must be made at run time based on the state of the processes and their location in the net.

A scheme for doing this based on names visible in some region of a distributed program was developed. Such names serve as rendezvous points between

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A174600

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 86-0783
6a. NAME OF PERFORMING ORGANIZATION: State University of New York; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: AFOSR
6c. ADDRESS (City, State and ZIP Code): Stony Brook, NY 11794-4466	7b. ADDRESS (City, State and ZIP Code): Bldg. 410, Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-81-0197
8c. ADDRESS (City, State and ZIP Code): Bldg. 410, Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2304	A2	

11. TITLE (Include Security Classification): The Design and Implementation of a Network Computer

12. PERSONAL AUTHOR(S): Arthur J. Bernstein

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 810615 TO 851214	860529	8

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The research performed under this grant is concerned with distributed languages and algorithms. Accomplishments are: (1) the design of a distributed language which has the property that the modules of a program written in it can be distributed over the modes of a network computer; (ii) the implementation of communication structures for supporting message passing; and (iii) the implementation of a two level system for supporting multicast message passing. In addition a theoretical study concerned with the application of verication techniques to real time programs was conducted. Four Ph.D theses, 3 Masters theses and 12 referred papers and conference proceeding were also produced.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Dr. Robert Buchal	767-5027	NM

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

processes through which messages can be transmitted. If the message is sent in unicast mode then any process within the region is eligible to receive it; if it is sent in multicast mode then multiple processes in the region may copy the message. Thus, name based addressing naturally integrates both concepts. Implementation of these ideas is described in (D).

The initial work on name based addressing formed the basis of a PhD thesis by Dr. David Gelernter, who is now on the faculty of the Computer Science Department at Yale. A paper describing this effort has been published [GB82]. An equipment grant from the National Science Foundation was obtained based on this work and another network related project in the department. Work on the project was continued by Dr. Mustaque Ahamad, and formed the basis of his PhD thesis. He is now on the faculty of Georgia Institute of Technology. Results on the uses and implementation of multicast were published in [AB85a].

B. Distributed Algorithms

A number of distributed algorithms of the type described above have been investigated.

1. Distributed Stable Storage - We have developed a distributed stable storage algorithm in which copies of a replicated database are distributed over nodes connected to a broadcast medium. This is a generalization of the standard stable storage model (Lecture Notes in Computer Science, vol 105, Distributed Systems - Architecture and Implementation, Springer Verlag, ch 11) in which data is duplicated on independent storage devices at a single node There are two disadvantages of the standard approach. Although data may be preserved at a node where a processor crash has occurred, it will not be available during the time that the processor is down. Secondly, malfunctions of the processor other than simple crashes can destroy the data.

The algorithm we have developed is designed specifically for data reliability in a broadcast environment. It is loosely coupled in the sense that it is designed to function despite the fact that not all copies of the data need agree and not all processors may be functioning correctly. A significant aspect of the proposal is that if no errors occur the redundancy is largely transparent to the procedures for accessing the data, requiring no extra communication. Additional messages are required only when errors are detected in copies of stored information. The system can handle a much wider class of errors than are handled by standard stable storage. A significant part of the work is the development of a Markov model to describe the failure behavior of the system. The model relates various parameters of the algorithm to the mean time to data loss. A paper describing this work has been published [Be85].

2. Distributed Dictionaries - We have developed an algorithm for updating replicated databases where serial consistency is not required. In the application we have considered, the database is a dictionary which supports the insertion and

deletion of entries. We have improved upon some earlier work in this area (Fischer and Michael, "Sacrificing Serializability to Attain High Availability of Data in an Unreliable Network", Proc. ACM Symp. on Principles of Database Systems", Mar 1982) by showing how communication costs can be reduced and the algorithm tailored to the topology of the network. We have developed several extensions to this model. One, which we refer to as the multiple insertion case, allows a given dictionary entry to be inserted more than once. In the original formulation of the problem this is prevented by tagging each entry with a unique number, thus forcing each entry to be unique. This may not be appropriate in applications where the creation of truly identical entries is unavoidable, the result of the delays involved in propagating information through a network. A second extension is a technique for reducing the size of data structures when the network becomes very large.

The work in this area formed the basis of a PhD thesis by Dr. Gene Wuu, who is now on the technical staff at Bell Telephone Laboratories. A paper describing the algorithm and a proof of its correctness was presented at a recent conference and was later selected to be reprinted in a journal [WB84].

3. Distributed Transactional File System - A major new research direction in this area is the study of distributed algorithms involved in implementing a distributed file system. Our interest in these algorithms is prompted by the recent award by the National Science Foundation of a Coordinated Experimental Research Grant to the Computer Science Department. (Stony Brook was ranked first in the nation in this award competition.) The proposed research involves building a system in which a relational database plays a central role. As one of the principal investigators on this grant my concern has been with the distributed nature of the system and is an outgrowth of AFOSR supported research. The work under AFOSR support has been concerned with developing distributed algorithms to support naming of relations and concurrency control for transactional access to relations in a network. In the naming area we have developed a technique to extend the UNIX file structure so that relations distributed across the network can be accessed in a transparent way from any site. The goal is similar to that of the LOCUS system but the approach we have taken involves no modifications to UNIX. The stress has been on developing an algorithm which imposes no additional overhead if the relation to be accessed is stored locally. A technique for dealing with name collisions is an important aspect of the work.

We have developed an optimistic algorithm for controlling concurrent access to a distributed database based on the original single site optimistic algorithm (Kung and Robinson, "On Optimistic Methods for Concurrency Control, ACM Trans. on Database Systems, June 1981). Transactions are numbered and each tuple in a relation is tagged with the number of its creator and (if it has been deleted) its deletor. In this way different versions of a relation, corresponding to different points in an equivalent serial order, can be extracted. By choosing the correct version it is possible for each transaction to see a consistent view of the database without resorting to locking and thus validation of read only

transactions is eliminated. This is a major advantage of the approach since in many applications read only transactions are considerably more numerous than transactions which update the database. The latter transactions must still be validated, but generalized validation conditions have been developed which substantially reduce the probability of rollback. An important part of the work is a technique which allows a transaction to extract the appropriate version simply. The algorithm is so structured that deadlocks and cascaded rollbacks cannot occur.

This work was initiated under AFOSR support and is now continuing under NSF support. The initial algorithm was described at a recent conference [ABGS86a] and a more comprehensive paper has been invited for publication in a journal [ABGS86b].

C. Network Communication

An implementation of the communication structures described in (A) motivated an investigation of novel technologies for supporting message passing. A technique called staged circuit switching (SCS) was developed which combines advantages of both the circuit switching and packet switching approaches. While circuit switching has a potential for high throughput, the initial delay in setting up a connection is a disadvantage, particularly when message lengths are small. In packet switching resources are acquired and released dynamically as the message travels through the network. Unfortunately, throughput for long messages is a problem since each packet must separately acquire the resources that it needs. In both cases a store and forward technique is generally used; the message is copied into the memory of each intermediate node along its path.

The SCS protocol attempts to set up a path, as in circuit switching, by reserving resources in advance. As soon as a situation arises in which a request for a resource cannot be granted the message is transmitted over the portion of the path that has been previously established and the resources used in that portion can be released. Thus, unlike circuit switching, SCS is non-blocking in the sense that the protocol never holds resources while waiting for other resources. Under heavy traffic situations this reduces to a packet switching protocol since resources will not be available for more than one hop at a time. In lightly loaded situations SCS reduces to circuit switching since blocking situations in the latter are unlikely in such circumstances.

The SCS proposal goes beyond a protocol in that hardware support was also designed. A small crossbar switch local to each node and under its control was proposed. Inputs to the switch consist of lines connected to the switches of physically neighboring nodes as well as a connection to the associated node. Thus a node can cause a direct connection to be made through its switch connecting the switches of two neighboring nodes. In this fashion the SCS protocol can establish a physical circuit from a source to a destination, thus avoiding the recopying of information at intermediate nodes. This has the effect of dynamically altering

the topology of the net since the set of nodes directly connected to a given node - its logical neighbors - can be varied. This in turn has implications with regard to the problem of how modules should be distributed in the net to optimize run time efficiency. Thus SCS allows experimentation with protocols in which heavily used message paths can be supported dynamically by physical connections.

The results of this investigation have been published [AGBB83]. The work was carried out by Dr. Gelernter and two Master's students, Mr. Mauricio Arango and Mr. Todd Morgan.

D. Support of Multicast

A two level system for supporting multicast message passing has been developed and implemented. At the higher level a name based addressing scheme, as described in (A), has been incorporated into Modual-2. A preprocessor for the language was built which recognizes communication statements and related data structures, and library modules are provided for supporting the new communication features at run time. The preprocessor will perform type checking related to interprocess communication and then convert programs using name based addressing into standard Modula-2 programs which call upon library modules to support communication. All the processes to which a particular name is visible consititute a multicast group, and any or all of them may receive a copy of a message addressed to the name.

At the lower level, a multicast protocol (MP) has been embedded in UNIX which is integrated into the 4.2 bsd socket structure. Multicast sockets operate in datagram mode with the difference that a set of processes (the multicast group) may essentially be connected to the same socket and thus receive copies of each message sent to the socket. MP is designed to utilize the internet protocol (IP) in a manner analagous to the implementation of TCP. Multicast addresses on the ethernet are associated on a one-to-one basis with names in the distributed language. All nodes in the net which support processes in the multicast group respond to the address corresponding the the associated name.

A version of MP has been implemented for use on a single ethernet. This work was carried out by Dr. Mustaque Ahamad with the help of a Master's student, Mr. Richard Trommer. A paper describing this project was presented at a recent conference [AB85b]. The protocol currently supports the distributed file system work described in (B.3) and has also been put into use at the University of Rochester. The preprocessor and library modules have also been implemented.

In extending the protocol for use in an internet environment two approaches can be taken. The simplest is one in which the sending node unicasts a copy of the message to be sent to some representative node on each net in the internet containing a process in the multicast group. Each representative then multicasts the message to members of the multicast group on its net using the above scheme. This suffers from the inefficiency that, particularly in large internets

with large multicast groups, duplication of unicast messages may result. This follows from the fact that distinct copies must be sent to each net even though they may follow essentially the same route. This can be avoided by using a more elaborate scheme, which we call extended multicast, in which a tree is dynamically maintained in the internet using a distributed algorithm and serves as a routing structure for delivering the message to be multicast to each net. This work has been described in recent publications [FWB84], [FWB85].

Another aspect of the work is concerned with developing an algorithm for maintaining the membership list of the multicast group in an internet when processes can enter and exit the group at run time. Message delays and network failures can cause inconsistency among the replicated copies of the list maintained by group members. A robust algorithm based on the dictionary algorithm described in (B.2) has been developed and is described in [FBW85].

E. Program Verification

The language study described above was complemented by more theoretical work concerned with the application of verification techniques to real time programs. Such programs are used to monitor or control events in an external environment and therefore must respond within certain hard time constraints. Processes, which are the asynchronous entities within the program, are natural sites for the code which handles the asynchronous events.

It is generally agreed that real time programs are the hardest to debug due to the difficulty of controlling the external environment and the unpredictability of the sequence and timing of events. Formal verification techniques, which have been developed for sequential programs and more recently adapted to concurrent programs, do not deal with real time issues. The purpose of this part of the research was to extend these techniques to include real time.

Temporal logic was found to be a suitable formal system upon which to base such extensions. Such a logic is similar to a standard first order logic with the addition of new temporal operators. A typical first order predicate may assert that something is true in the current state. This can be modified to a temporal assertion which deals with the truth of a predicate in the current and all future states. Alternatively, a temporal assertion may claim the ultimate truth - at some future time - of a predicate.

The approach taken in our work is to extend temporal logic to include assertions in which time is explicitly included. Thus, asserting that a particular state will or will not be reached within the next t time units is possible. In order to do this new axioms and rules of inference dealing with time are introduced into the model. In the resulting formal system it is possible to make assertions about the ordering of states as execution unfolds based on known time constraints on events in the external world. Using this model we are able to prove some interesting properties of real time programs (e.g., a handler process will always be ready when the next event occurs, a timeout situation cannot occur).

A paper describing this work was published [BH81]. The research served as the basis of a PhD thesis for Dr. Paul Harter who is now on the faculty of the Computer Science Department at the University of Colorado. A grant was also obtained from the National Science Foundation to continue the work.

Since the language work is directed towards supporting low level distributed algorithms, timeout plays a significant role. As a separate project we have studied the semantics of timeout to develop a formal verification technique for message passing programs in which a sender or receiver of a message may timeout if message transmission is delayed beyond a specified interval. Axioms for describing such communication statements are developed. An important issue here is the notion of predicate transfer between the communicating processes. The predicate describes the information which can be deduced by the process which has timed out about the state of its correspondant. It is shown that even when no message is exchanged, information can be transferred between processes. A paper describing this work has been published [Be86].

References

[BH81] P. Harter and A. J. Bernstein, "Proving Real Time Properties of Programs With Temporal Logic", 8th Symp on Operating Systems Principles, ACM, Pacific Grove, Cal., Dec 1981.

[GB82] D. Gelernter and A. J. Bernstein, "Distributed Communication via Global Buffer", ACM SIGACT-SIGOPS Symp on Principles of Distributed Computing, Ottawa, Canada, Aug 1982.

[AGBB83] M. Arango, D. Gelernter, H. Badr and A. J. Bernstein, "Staged Circuit Switching for Network Computers", ACM SIGCOMM 83 Symp: Communications, Architectures, and Protocols, Austin, Texas, Mar 1983.

[FWB84] A. Frank, L. Wittie and A. J. Bernstein, "Group Communication on Netcomputers", 4th Int'l. Conf. on Distributed Computing Systems, San Francisco, CA., May 1984.

[WB84] G. Wuu and A. J. Bernstein, "Efficient Solutions to the Replicated Log and Dictionary Problems", 3rd ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing, Vancouver, Canada, Aug 1984 (selected by SIGOPS for republication in Operating Systems Review, Jan. 1986).

[AB85a] M. Ahamad and A. J. Bernstein, "The Application of Name Based Addressing to Low Level Distributed Algorithms", IEEE Trans. on Software, vol SE-11, Jan 1985.

[Be85] A. J. Bernstein, "A Loosely Coupled Distributed System for Reliably

Storing Data", IEEE Trans. on Software Engineering, vol SE-11, May 1985.

[FWB85] A. Frank, L. Wittie and A. J. Bernstein, "Mulitcast Communication in Distributed Groups on Network Computers", IEEE Software, vol 2, May 1985.

[AB85b] M. Ahamad and A. J. Bernstein, "Multicast Communication in UNIX 4.2", 5th Int'l Conf. on Distributed Computing, Denver, Colorado, May 1985.

[WB85] G. Wuu and A. J. Bernstein, "False Deadlock Detection in Distributed Systems", IEEE Trans on Software Engineering, vol SE-11, Aug 1985.

[FBW85] A. Frank, A. J. Bernstein and L. Wittie, "Maintaining Weakly-Consistent Replicated Data on Dynamic Groups of Computers", 14th Int'l. Conf. on Parallel Processing, St. Charles, IL, Aug. 1985.

[ABGS86a] D. Agrawal, A. J. Bernstein, P. Gupta and S. Sengupta, "Distributed Multi-Version Optimistic Concurrency Control for a Relational Database System", CompCon '86, San Francisco, CA, Mar 1986.

[Be86] A. J. Bernstein, "Predicate Transfer and Timeout in Message Passing Systems", to appear, Information Processing Letters, 1986.

[ABGS86b] D. Agrawal, A. J. Bernstein, P. Gupta and S. Sengupta, "Generalized Optimistic Concurrency Control", invited for publication, Distributed Computing, Springer-Verlag, 1986.

END

1-87

DTIC